Chapter one and two of the Lex and Yacc book

# Luis Ricardo Espinoza Rodriguez A01229141

## The simplest lex program

The book starts by examining what probably is the simplest lex program. What it does it scans for any input it prints it. The book references the *cat* command, saying that this program would imitate such command if no other flag was given. The regular expression used is the following

**. | \n**

This expression has two parts divided by the pipe which acts as a logical OR. A dot represents any and all characters except for new line. The other part is the new line. This effectively covers all possible characters.

## Recognizing words with lex

This exercise is very simple. The book shows a very small example of how to detect certain words using lex. On this exercise there are not many regular expressions used. The book focuses on showing that lex checks for ambiguity by default. It matches a word only once and it matches it whit the rule that has the most restrictions. That’s why at the end of the code there is the following regex.

**[a-zA-Z]+**

This expression matches any single character word. So, any word that is not ruled out before is going to end in this expression.

## Grammars

Then the book explains how the scanner communicates with the parser. You can communicate between lex and yacc by returning values when you encounter a string that matches a rule. This returned value is going to be cached by yacc. For this, both files have to have the tokens, so that the values that are returned by lex can be understood by yacc.

Then the books explains a simple yacc program to parse an English sentence. It is mostly the same but the rules part instead of using regular expressions uses *grammars*.

## Using Lex

In this chapter the book explains some lex programs. First it explains some of the features that regex has. It then shows some examples of common regex rules. Then the first program that it introduces is a word counting program. The next program is to parse the input of a command line.